·创新创业技术·

# 基于可编程数据平面的实时传输技术研究

董智健，蔡文龙，汤泽康，周　舟，陈美娟*

（南京邮电大学 通信与信息工程学院，南京 210003）

**摘要：**随着人们线上业务的增多，在线教育、在线会议等业务对网络时延和抖动有了更高的要求，因此，该文基于软件定义网络(SDN)架构，在可编程数据平面使用 P4 语言设计了一种实时传输方案。该方案基于可变阈值机制，通过在可编程数据面上建立一种阈值反馈自调节机制，使优先级调度和重路由调度能适应网络波动，降低时延。实验结果表明，改进后方案在时延和抖动方面均有明显改善，提高了实时传输业务质量。

**关　键　词：**软件定义网络；P4 语言；实时传输；可编程；重路由

**中图分类号：**TN91　　**文献标志码：**A　　DOI: 10.12179/1672-4550.20220523

# Research on the Real-time Transmission Technology Based on the Programmable Data Plane

DONG Zhijian, CAI Wenlong, TANG Zekang, ZHOU Zhou, CHEN Meijuan*

（School of Communication and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China）

**Abstract:** Due to the impact of the new crown epidemic, people's online services have increased. Among them, online education, online conferences and other services have higher requirements on network delay and jitter. Therefore, based on the software defined network (SDN) architecture, this paper designs a real-time transmission scheme using the programming protocol-independent packet processors (P4) language in the programmable data plane. The scheme is based on a variable threshold mechanism and establishes a threshold feedback self-adjustment mechanism on the programmable data plane, so that priority scheduling and rerouting scheduling can adapt to network fluctuations and reduce latency. The experimental results show that the improved scheme has obvious improvements in delay and jitter, and improves the quality of real-time transmission services.

**Key words:** SDN; programming protocol-independent packet processors language; real-time transmission; programmable; rerouting

随着通信技术的不断发展，人们办公方式逐渐转变，线上业务逐渐增多，人们对于传输时延的要求也逐渐提高，而传统的网络架构已经不能满足人们的需求。为满足这些需求，斯坦福大学的研究人员在 2006 年提出了软件定义网络(SDN)的概念[1]。开发人员可以通过在 SDN 架构中编写的动态和自动化程序，轻松快速地管理、配置和优化网络资源。此外，由于 SDN 采用集中式控制面，SDN 网络的控制器可以观察整个网络，使得控制器具有实时优化、管理、改变和修改流程和资源的能力。目前 SDN 控制转发平面往往采用 OpenFlow 协议[2]，但 OpenFlow 是基于协议的，要实现新的功能必须不断地打补丁加扩展，使得 OpenFlow 越来越庞大。为了应对 OpenFlow 因自身设计带来的可扩展性差的问题，McKeown 等人提出的可编程协议无关报文处理语言(P4)以

**收稿日期：**2022−08−31；**修回日期：**2024−04−18

**基金项目：**江苏省省级大学生创新创业训练项目 (SYB2021002)；南京邮电大学研究生教学改革项目(JGKT22_XYB12)。

**作者简介：**董智健(2000−)，男，本科，通信与信息工程专业。

*** 通信作者：**陈美娟(1971−)，女，博士，副教授，主要从事区块链与边缘计算结合的移动通信网络安全机制、SDN/NFV 的通信网络资源部署与优化、机器学习在移动通信网络性能优化等方面的研究。E-mail: chenmj@njupt.edu.cn

及相应的转发模型[3–4]，可以对转发设备的数据包处理行为进行定义，开发者可以使用支持 P4 语言的交换机灵活地自定义网络协议和相关处理逻辑。

目前在传统网络的研究上已经有了很多保障实时传输的方案[5–6]，比如资源预留协议(resource reservation protocol，RSVP)[7]；但 RSVP 协议在降低时延方面仍然具有一些问题[8]。为解决这个问题，文献 [9] 以 P4 为基础，设计了一套基于已花费时间和队列时延[10] 的优先级调度和重路由方案。该方案完全基于可编程数据平面实现，通过降低队列时延和重路由来保障实时传输；但是该方案的重路由部分均采用固定阈值，面对网络的剧烈波动以及变化巨大的数据流处理能力依旧不足。

因此，本文在文献 [9] 研究的基础上，基于可编程数据面采用反馈实现了重路由阈值的动态改变。在网络剧烈波动的情况下，通过动态地调节时延，实现合理的网负载均衡，进而保障时延。本文的贡献主要体现在对原方案的一些优先级级数设计的合理性进行了测试与分析，并通过对重路由方案进行改进，使得基于本文方案的实时传输技术更具有普遍性。

## 1　可编程数据平面相关技术

### 1.1　P4 程序流程

P4 语言是一种定义数据平面转发规则的高级编程语言。P4 与协议无关，可以实现对数据包层面的精细调控。因此，网络开发者或运营商能够根据应用需求将各种协议编译到支持 P4 语言的设备中，方便其他开发者进行新协议的开发与部署，具有良好的可扩展性。

P4 语言程序的工作流程如图 1 所示[11]。开发者会根据网络的需求定义转发设备的转发规则；接着对 P4 语言程序进行编译，生成 JSON 格式的配置文件，并载入数据平面的 P4 转发设备配置文件当中；与此同时，转发设备的配置文件与应用程序接口(application programming interface，API)会自动加载到数据平面转发设备中并更新解析器和匹配动作表；最后数据平面的 P4 转发设备会根据 P4 语言程序中的定义实现一系列操作。

本文设计的 P4 语言程序基于 v1model 交换机[12]，其结构如图 2 所示。解析器是一个有限状态机，定义了数据包头部的解析流程；入口流水线和出口流水线处执行各项流控制程序以及匹配动作表以实现相应的转发规则。

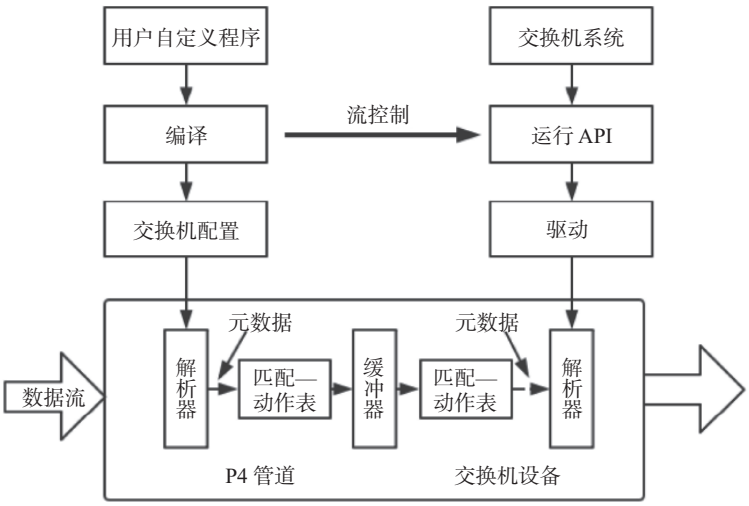


图 1　P4 语言程序的工作流程

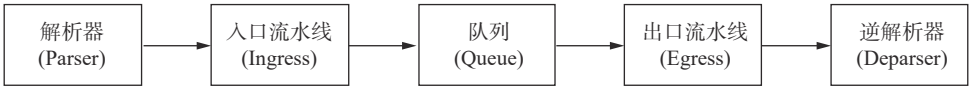


图 2　v1model 交换机结构图

### 1.2　端到端时延分析

在存储转发交换的网络中，数据包从一端到另一端总时延 $D$ 由 4 部分构成：处理时延 $D_{\text{process}}$、传输时延 $D_{\text{translate}}$、队列时延 $D_{\text{queue}}$ 和传播时延

$D_{\text{propagate}}$，即[13]：

$$D = D_{\text{process}} + D_{\text{translate}} + D_{\text{queue}} + D_{\text{propagate}} \quad (1)$$

其中处理时延和传输时延不做考虑。处理时延指的是交换机处理数据包所需的时间，其大小一般在微秒级别，在流表数目合理的情况下，相比其他时延，该时延可以暂时忽略。传输时延指的是将一个数据包所有数据传输到传播介质上所需的时间。由于每个数据包的总长度有限，传输时延的大小与线路本身性质有关：

$$D_{\text{translate}} = \frac{L_{\text{package}}}{R} \quad (2)$$

式中：$L_{\text{package}}$是数据包的总长度，$R$是传输速率。

本文主要考虑队列时延和传播时延。队列时延指的是数据包在队列中等待的时间，大小范围在微秒至毫秒级别：

$$D_{\text{queue}} = ID_{\text{translate}} \quad (3)$$

式中：$I$是队列平均长度；$D_{\text{translate}}$是传输时延。队列时延取决于链路的负载情况。由于传输时延很难直接提升，因此可以通过合理的优先级队列调度方法来改善队列时延。由于 v1model 支持队列，因此可以通过合理的优先级调度改变数据包在队列中的优先级来降低队列延迟。

传播时延是指电信号通过传输介质从一个节点传播到下一个节点所需的时间，大小取决于线路本身的性质，其值为：

$$D_{\text{propagate}} = \frac{L_{\text{line}}}{s} \quad (4)$$

式中：$L_{\text{line}}$是线路的长度；$s$是介质的传输速度。当然传播时延也不能改变，不过通常传输的链路会有两条及以上不同的路径，开发者根据带宽和传播时延决定一条更好的主路径。当主路径拥堵时，可以通过修改目的地址(mac)和发送端口将一些时延较高数据包切换到“辅道”。一方面让这些数据包尽快到达目的端；另一方面也分担了主链路的压力。该方法被称为重路由(rerouting)。

## 2　可变阈值重路由方案设计

### 2.1　基于已花费时间的可变阈值重路由方案设计

方案的最终目的是降低时延，从宏观的角度考虑，根据数据包在网络中传输的已花费时间选择其优先级，如图 3 所示，从 $S_3$ 到 $S_5$ 有两条路径可选，假设经过 $S_4$ 的路径为路径 1，经过 $S_6$ 的路径为路径 2。路径 1 带宽较大，但路径较长；路径 2 带宽很小，路径短。在这种情况下，为了传送大量的数据流，选择路径 1 作为主路径。网络拥堵情况严重时，可以采用重路由的方法，即将时延大的数据包重路由到路径 2 上，一方面缓解主路径的压力，另一方面路径 2 传输快，可以降低数据包的时延。对应的数据面流程图如图 4 所示。

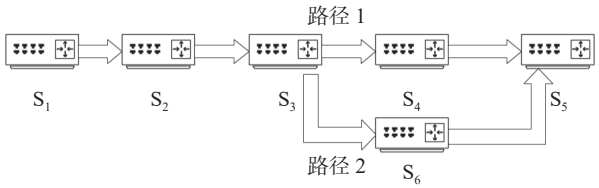


图 3　基于已花费时间的重路由示意图

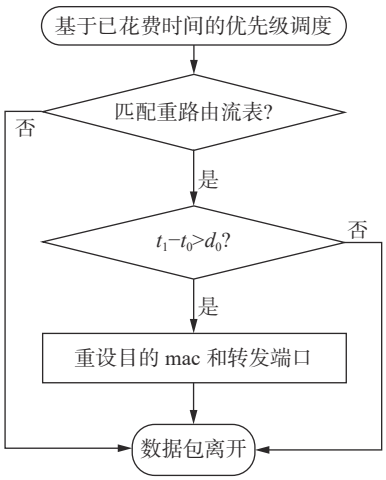


图 4　基于已花费时间的重路由方案流程图

在 $S_1$ 处记录当前时间$t_0$。在 $S_2$ 处，当数据包进入 ingress 阶段时，读取当前时间$t$，计算$t-t_0$。给定一个已产生时延的阈值$d_0$，当数据包到达 $S_3$ 时，在 ingress 阶段，当优先级调度结束后，将$t-t_0$与$d_0$对比，当$t-t_0>d_0$时，则执行重路由，将出端口转至路径 2，否则继续传输至路径 1。

### 2.2　基于队列延迟的可变阈值重路由方案设计

同 2.1 节情况下，为了传送大量的数据流，选择路径 1 作为主路径。该方案对应的数据面流程图如图 5 所示。

在 ingress 阶段，读取进入 ingress 阶段的时间$t_1$，在 egress 阶段读取进入 egress 的时间$t_2$，并将$t_2-t_1$写入总计队列时延$d_{\text{acc}}$中，跳数加 1。可以给定一个队列时延的阈值$d_0$，当数据包到达 $S_3$ 时，

在 ingress 阶段，当优先级调度结束后，将$t-t_0$与$d_0$对比，当$t-t_0>d_0$时，则执行重路由，将出端口转至路径 2，否则继续传输至路径 1。

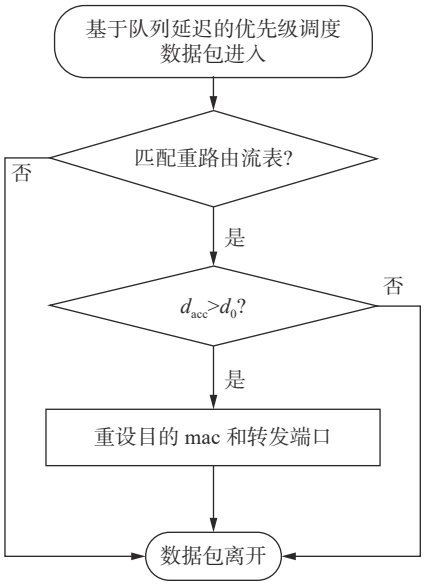


图 5 基于队列时延的重路由设计流程图

## 3 方案实现

### 3.1 可变阈值重路由整体方案

本文基于可编程数据面，由可编程交换机根据数据包的时延情况，设置反馈机制动态地调节阈值。设计一个计数器，记录保持上一状态的数据包个数，即记录连续有几个数据包均执行或均不执行重路由。如果记录的数据包个数达到某一限值，则执行反馈，对应增加或减少阈值，实现阈值随网络变化而动态变化。

### 3.2 已花费时间可变阈值重路由

本文使用一个专用匹配表用于接收控制面下发的各项数据，包括已花费时间阈值$\eta$，阈值增减变化值($\delta_1$, $\delta_2$)等。该表仅在执行 P4 runtime 时和其他流表一起下发，因此整体方案仍全部在可编程数据面内完成。

此方案在 ingress 阶段的主程序中通过比较阈值$\eta$和已花费时间$t-t_1$直接判断选择表项。当已花费时间连续 8 次高于阈值$\eta$执行重路由后，将阈值$\eta$增大$\delta_1$，将重路由的阈值提高，以让时延更大的数据包优先重路由。相反的，当已花费时间连续 8 次低于阈值$\eta$均不执行重路由后，判断数据包此时时延较低，重路由机制不起作用，将阈值$\eta$减小$\delta_2$，以让时延相对更大的数据包进行重路由。这里使用 3 位计数器。当计数器达到 8 时，由于溢出计数器会自动归零，通过判零即可得知是否需要执行反馈机制。

阈值增减量的取值合理性极大地与能否有效改进和改进多少息息相关。数据面流程图如图 6 所示，控制面流程图如图 7 所示。

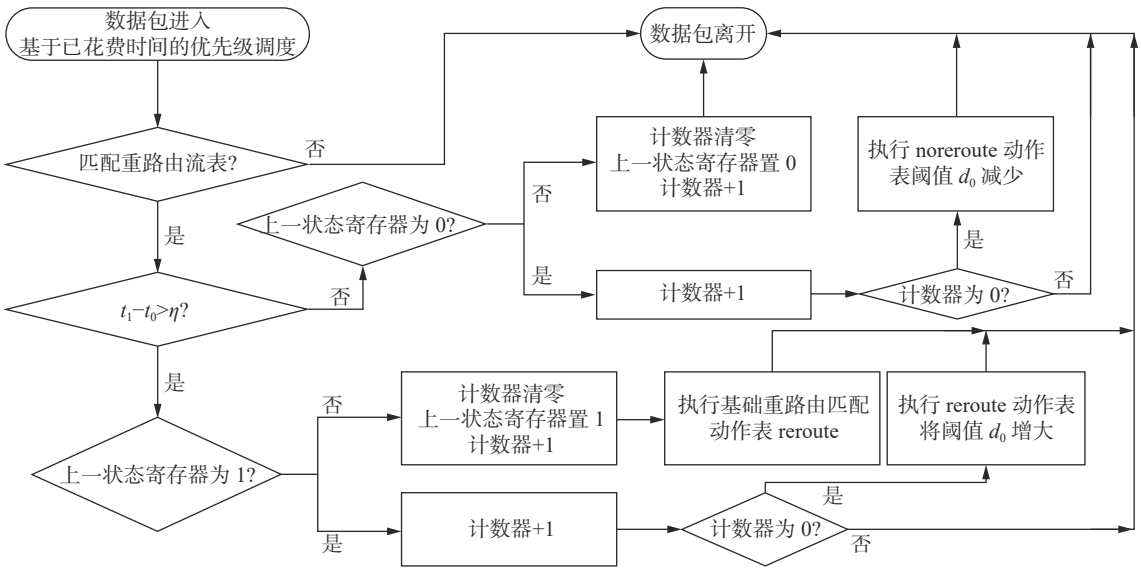


图 6 基于已花费时间的重路由改进方案流程图

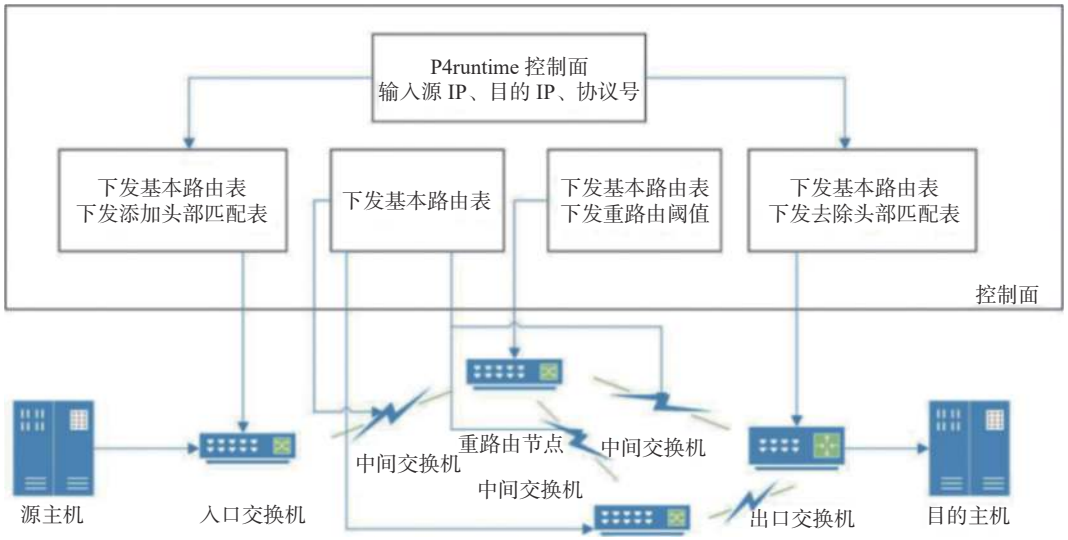


图 7　控制面流程图

### 3.3　队列时延可变阈值重路由

基于队列时延的方案和 3.2 节的方案基本一致。本文将队列时延阈值$\zeta$、阈值增减变化值($\delta_1$, $\delta_2$)等各项数据直接写在 P4 程序内，进一步减少了控制面的花销。

当累计队列时延$d_{acc}$连续 8 次高于阈值$\zeta$执行重路由时，判断数据包此时时延较高，故将阈值$\zeta$增大$\delta_1$，以让时延更大的数据包优先重路由。相反的，当连续 8 次低于阈值$\zeta$均不执行重路由时，将阈值$\zeta$减小$\delta_2$，以让时延相对更大的数据包进行重路由。数据面流程图如图 8 所示。

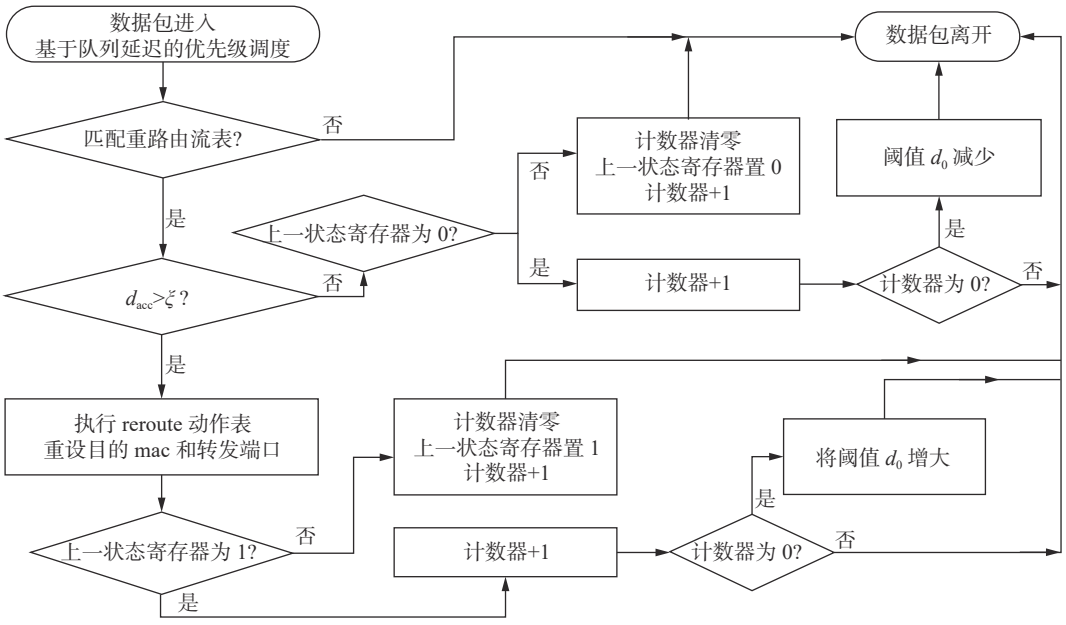


图 8　基于队列时延重路由流程图

## 4　性能测试

### 4.1　测试方案

本文使用 Mininet 和 P4 bmv2 虚拟交换机进行测试，采用的拓扑图如图 9 所示。该拓扑基于文献 [6] 采用的 Internet2 拓扑进行简化，将实验用到的部分提取出来，节约内存资源，便于实验。

实验的主数据流从 $h_1$ 发送到 $h_{13}$，从偶数号主机发送数据到下一个奇数号主机作为背景流。实验环境如表 1 所示。

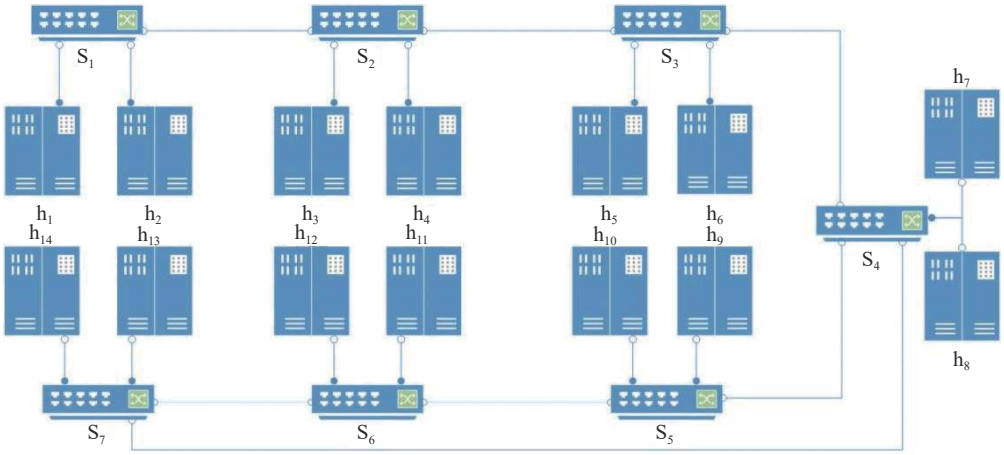


图 9 测试采用拓扑图

表 1 实验环境

| 设备 | 信息 |
|---|---|
| CPU | AMD Ryzen7 5800H |
| 内存 | 32G |
| 操作系统 | Ubuntu 20.04 |
| Mininet版本 | 2.3.0 |
| BMV2版本 | b447ac4c0cfd83e5e72a3cc6120251c1e91128ab |

为了还原对比(原)方案的实验情况，本文使用 Multi-Generator(MGEN)生成数据包间隔符合泊松分布的流量。使用抓包工具 wireshark 在源主机与目的主机处抓包计算时延。每次实验进行 160 s，去掉开始 20 s 和结束 20 s 的数据防止误差。每个测试进行 7 遍，去掉一个最大值，去掉一个最小值，剩下的取平均值作为实验数据。

### 4.2 固定阈值和可变阈值性能分析

采用已花费时间阈值 4 500，测试使用原方案改进后的情况，测试结果如表 2 所示。

表 2 已花费时间改进方案测试

| 已花费时间阈值 $\eta$/μs | 阈值变化量 $(\delta_1,\delta_2)$/μs | 时延/ms | 抖动/ms |
|---|---|---|---|
| 4 500 | — | 4.710 | 0.853 |
| 4 500 | 300 | 4.684 | 0.867 |

由表 2 可知，从数据上看改进方案时延相比原方案有了明显的降低，但抖动有所提升。本次实验采用泊松分布数据流，该数据流服从泊松分布，会出现一个较大的流量攀升阶段。因此，固定阈值会导致数据包在此阶段全部重路由，时延不会得到改善。

采用队列时延阈值 1 500，测试改进方案的情况，结果如表 3 所示。可以看出，改进方案无论是时延还是抖动相比原方案均有所改进，说明改进方案取得了预期的效果。由于数据流服从泊松分布，在流量陡然上升的阶段队列时延会很高。基于固定阈值会使得数据包在此阶段全部重路由，时延不会得到改善；而改进方案采用可变阈值，在此阶段阈值会随之增加，进而能够合理地实现重路由机制。

表 3 队列时延改进方案测试结果

| 队列时延阈值 $\zeta$/μs | 阈值变化量 $(\delta_1,\delta_2)$/μs | 时延/ms | 抖动/ms |
|---|---|---|---|
| 1 500 | — | 4.554 | 0.817 |
| 1 500 | 250 | 4.445 | 0.805 |

### 4.3 基本阈值变化量性能分析

为确定阈值变化量对时延和抖动性能的影响，取阈值变化量 100~1 000 间 6 个值进行测试，结果如表 4 所示。

表 4 已花费时间重路由阈值变化量测试

| 已花费时间阈值 $\eta$/μs | 阈值变化量 $(\delta_1,\delta_2)$/μs | 时延/ms | 抖动/ms |
|---|---|---|---|
| 4 500 | — | 4.710 | 0.853 |
| 4 500 | 1 000 | 5.030 | 0.894 |
| 4 500 | 500 | 4.754 | 0.850 |
| 4 500 | 300 | 4.684 | 0.867 |
| 4 500 | 250 | 4.651 | 0.831 |
| 4 500 | 200 | 4.698 | 0.837 |
| 4 500 | 100 | 4.808 | 0.872 |

由表 4 可知，当阈值变化量为 200 和 250 时，时延和抖动性能均好于原方案。故已花费时间阈值$\eta$的取值和阈值变化量($\delta_1$，$\delta_2$)取值合理与否均对时延和抖动具有控制作用。当取值合理时，取得的效果也就越好；当取值不合理是，会影响重

路由的判断。由表 4 可知，合理的取值区间应该在变化量可取值范围的中值区域。

可变阈值存在一个上升下降率，即阈值变化的速率的问题，上升下降率与阈值的变化量成正比。但由于阈值的变化存在延迟，所以可能导致时延的数据包被重路由，高时延的数据包不被操作。由表 5 可以看出，当阈值变化量取值在 100~500 之间时，取值相对合理，故时延和抖动均好于原方案值。

表 5　队列时延方案中阈值变化量$\delta_1,\delta_2$的影响

| 队列时延阈值 $\zeta$/μs | 阈值变化量 $(\delta_1,\delta_2)$/μs | 时延/ms | 抖动/ms |
|---|---|---|---|
| 1 500 | — | 4.554 | 0.817 |
| 1 500 | 750 | 4.562 | 0.856 |
| 1 500 | 500 | 4.347 | 0.793 |
| 1 500 | 250 | 4.445 | 0.805 |
| 1 500 | 100 | 4.520 | 0.808 |

综上，队列时延阈值$\zeta$的取值和阈值变化量$(\delta_1,\delta_2)$取值合理与否对时延和抖动均具有不可忽视的影响。当取值合理时，取得的效果也就越好；反之则会影响其原本的性能，得不偿失。合理的取值区间应该在变化量可取值范围的中值区域。

## 5　结束语

基于传统网络构架不能满足人们对传输的要求，本文基于 SDN 架构，在可编程数据平面使用 P4 语言设计了一种实时传输方案。采用可变阈值机制，通过在可编程数据面上建立一种阈值反馈自调节机制，使优先级调度和重路由调度能适应网络波动，降低时延。本文通过对固定阈值和可变的性能分析、基本阈值变化的性能分析得出，基本阈值的取值区间在阈值变化量的可取值范围中值区域时对时延和抖动具有较好的控制作用，阈值变化量在 100~500 之间，对延时和抖动的控制效果较好。

## 参 考 文 献


[1] MCKEOWN N. Software-defined networking[C]// Proceedings of 28th IEEE Conference on Computer Communications(INFOCOM2009). Rio De Janeiro: IEEE Press, 2009.

[2] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. Openflow: Enabling innovation in campus networks[C]//ACM SIGCOMM Computer Communication Review. New York: Assoc Computing Machinery, 2008.

[3] BOSSHART P, DALY D, GIBB G, et al. P4: Programming protocol-independent packet processors[C]//ACM SIGCOMM Computer Communication Review. New York: Assoc Computing Machinery, 2014.

[4] 杨爱玲, 邹乾友, 付松涛. P4 交换机在天地一体化网络中的应用[J]. 信息工程大学学报, 2020, 21(4): 453−458.

[5] 戴帅, 肖楠, 梁俊, 等. 基于处理时延的卫星网络 TCP 拥塞控制算法[J]. 现代防御技术, 2014, 42(3): 127−134.

[6] 李元青. 移动通信中多径传播时延的分布密度[J]. 电子学报, 1986(2): 92−99.

[7] RFC 2205. Resource reservation protocol (RSVP): Version 1 functional specification[S]. USA: IETF REC. 1997.

[8] 翁惠玉, 刘芳, 陈志英, 等. Intserv/RSVP 的现状及存在的问题[J]. 数据通信, 1999(4): 4−7.

[9] 张宇航. 基于 P4 的实时传输技术研究[D]. 杭州: 浙江大学, 2020.

[10] 吴东. 控制时延的主动队列管理算法[J]. 计算机应用, 2014, 34(3): 632−634 .

[11] 林耘森箫, 毕军, 周禹, 等. 基于 P4 的可编程数据平面研究及其应用[J]. 计算机学报, 2019, 42(11): 2539−2560.

[12] P4 Language Consortium (2015) The P4 Language Specification, Version 1.0.2[EB/OL]. [2019-11-15]. https://p4.org/specs/.

[13] 侯玮, 窦睿彧, 兰巨龙. FAST TCP 拥塞控制机制研究[J]. 微计算机信息, 2005(8): 10−11 .


编辑　张俊